DEACON BREADBOARD SUMMARY,

GENERAL ELECTRIC CO SANTA BARBARA CA

MAR 1964

The body of this report is divided into three parts. First, the general environment of DEACON, the general objectives which give meaning to the DEACON Breadboard, the embodiment of DEACON in program and computer, and the next steps beyond the Breadboard are discussed. Second, the most significant technical advance offered in DEACON—the use of semantic transformations of data in direct correspondence with syntactic manipulations—is discussed. Third, the organization of the Breadboard's program structure (how it does the job) is outlined. The appendix contains a listing of the current rules of grammar and a tabulation of the data base.

DEACON rests on a number of previous results. The work of Dr. Robert Lindsay on SAD SAM,[1,2] as well as his invaluable discussions on DEACON itself, played a significant role. The ideas underlying Baseball,[3,4] originating with Dr. Bert Green, et al., were suggestive and supported the feasibility of the goal. Modern research in structural linguistics, particularly the papers of Chomsky[5,6] and Yngve,[7] has provided important concepts. These various ingredients to our work come from the recent advances in computational linguistics. It was the infusion into this line of thought of the fundamental work of Dr. Alfred Tarski in mathematical semantics[8] that supplied the central key to DEACON techniques.

The authors wish also to acknowledge the criticism and innumerable suggestions of Mr. Jerry Kindred and Mr. Robert Callan. The contributions of Mr. Harold McBeth in the underlying LAP list-processing language and also in early conversations concerning DEACON techniques also played an essential role in the early stages of the work.

The body of this report is divided into three parts. First, the general environment of DEACON, the general objectives which give meaning to the DEACON Breadboard, the embodiment of DEACON in program and computer, and the next steps beyond the Breadboard are discussed. Second, the most significant technical advance offered in DEACON—the use of semantic transformations of data in direct correspondence with syntactic manipulations—is discussed. Third, the organization of the Breadboard's program structure (how it does the job) is outlined. The appendix contains a listing of the current rules of grammar and a tabulation of the data base.

DEACON rests on a number of previous results. The work of Dr. Robert Lindsay on SAD SAM,[1,2] as well as his invaluable discussions on DEACON itself, played a significant role. The ideas underlying Baseball,[3,4] originating with Dr. Bert Green, et al., were suggestive and supported the feasibility of the goal. Modern research in structural linguistics, particularly the papers of Chomsky[5,6] and Yngve,[7] has provided important concepts. These various ingredients to our work come from the recent advances in computational linguistics. It was the infusion into this line of thought of the fundamental work of Dr. Alfred Tarski in mathematical semantics[8] that supplied the central key to DEACON techniques.

The authors wish also to acknowledge the criticism and innumerable suggestions of Mr. Jerry Kindred and Mr. Robert Callan. The contributions of Mr. Harold McBeth in the underlying LAP list-processing language and also in early conversations concerning DEACON techniques also played an essential role in the early stages of the work.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS AND TABLES

SECTION 1

INTRODUCTION

BACKGROUND

The computer has been considered primarily as an instrument for carrying out vastly complex calculations and for performing repetitive operations on large bodies of data. However, there have been severe limitations in our ability to communicate with it. These limitations are not restrictive when the ratio of input and output to internal calculation is very small. For example, in the solution of a differential equation governing some natural phenomenon, the input of the few parameters of the equation is enough to initiate literally tens of millions of calculation steps, which in turn may output only a few hundred numbers. Similarly, in the use of the computer for routine data processing, such as payroll preparation or other batch processing of records, the necessary instructions for the computer need be written only once. Therefore, existing methods of using computers in such applications have seemed to be efficient.

However, when we turn to the use of computers in, say, management information systems or military command and control, we are faced with an entirely different situation. The very essence of these applications is the capability to bring to bear, on an unending and constantly evolving variety of situations, underlying data describing the state of affairs. Change and the unanticipated are relevant in two ways to such applications:

First, the questions that one wishes to ask the system are always new. It is seldom that we wish to know a single fact that is on file. Furthermore, the busy executive has no time to put together the relevant evaluation from a mere retrieval of ingredient facts, and lengthy listings of material may only compound his frustration.

Second, and of even greater potential significance, is the fact that the relevance of a given view of the world changes constantly. The items of data in a data base, and the way that these items are structured in the base, reflect a certain view of the world. This view is but one of myriad possible views; the relevance of this

> particular view to the current situation determines to a large extent the degree of control that an organization can exercise over its environment, and this relevance is a very volatile thing. Informed observation of any vital organization discloses a surprisingly high rate of evolution in its view of itself and its environment, although we overlook this fact in so many of our current information-systems designs. Indeed, we would go so far as to suggest that the degree of control that is exercised by an organization is in direct proportion to the dynamics of its data base.*

Because of these two aspects of change, information systems for management and control functions must have the following characteristics:

- They must have the ability to respond to queries and instructions that are both unanticipated and complex in both their retrieval and calculation aspects.
- They must be compatible with constant evolution of the content and structure of the underlying base of data and of routines that are specific to the data base. Such evolution can be adequate only if it derives essentially as a matter of course from the users' routine use of the system.

The technological difficulties in realizing these two capabilities can best be discussed in terms of language. Natural language—English, in the DEACON Breadboard—provides great flexibility of expression. It also provides the media by which new expressions may evolve, increasing the efficiency of our communication. We subsume very complex and extensive definitions under such phrases as "financial statement," "budget," and "labor turnover." Often, as such words achieve special status in the conversations of a tightly knit team, they express exacting and precise ideas of great complexity. It is the evolution of such words and of their specialized meanings that is important here. And underlying this evolving vocabulary is the versatile syntax of English that can be used in such definitions, permitting the expression of the novel and the unanticipated by which we have characterized such control environments.

Finally, turning back to scientific and bulk-processing type applications of computers, we are beginning to realize that the same processes

*For a more detailed discussion of the concept summarized in this paragraph, see References 9 and 10.

of change also result in the decay of what we believed were current computer systems efficiencies. The language flexibilities that are necessary in control situations add dimensions to the applicability of computers in scientific and data processing situations which bring astounding efficiencies over current system practices when imaginatively applied (see Reference 11, for example).

DEACON OBJECTIVES

Based on this background, objectives of the DEACON Project were formulated for developing a computer system that will

- respond directly to queries and instructions phrased in natural language
- program itself through its ability to understand the semantic content of English syntax and thereby carry out both the retrieval and calculation aspects required in responding to an unanticipated query
- be independent in its operation from the content, extent, and organization of data in memory, and permit modification of that material through direct instruction
- permit the direct addition of new vocabulary, either by definition in terms of words already understood, or by addition of data and their corresponding names.

In these statements of desired capabilities, the word "direct" means that the user, i.e., executive or staff, has access to the computer from his normal working area with turnaround time from query to answer of about a minute.

Certain aspects of these objectives are already being realized on current systems and, therefore, their feasibility can be assumed. In particular, direct access to computers from remote consoles, with several such consoles in use simultaneously, is being realized in several installations.* Input/output devices are adequate for present purposes and much work is being done elsewhere to improve both the means of communicating input to the computer and of displaying required output. Other hardware aspects are similarly adequate now and are undergoing extensive development. Also, an adequate

*At the Systems Development Corporation and in Project MAC at the Massachusetts Institute of Technology, both under sponsorship of the Advanced Research Projects Agency.

background for dealing with the syntactic aspects of the language has been provided by current research in theoretical linguistics.

In the semantic aspects of computational linguistics, however, previous research appears inadequate for accomplishing the above objectives. Consequently, the principal problem that must be solved is that of the semantic interface: How is the computer to be programmed so that it can, to a reasonable degree, understand the meaning of sentences communicated to it? The test of adequacy is clear: namely, the computer must be able to respond in a sensible manner.

It is to this semantic interface problem that we have given theoretical attention,[12] and it is this problem that we believe we have solved sufficiently for meeting the DEACON objectives. Establishing the adequacy of the solution can be done only by constructing a system that understands and responds to English sentences of sufficient syntactic complexity and semantic interest to demonstrate the feasibility of a fully implemented DEACON system. We believe the DEACON Breadboard to be such a system.

THE DEACON BREADBOARD

The Breadboard consists of a set of programs and data for the GE 225 computer. The GE 225 is a small-to-medium-sized computer, which has a cycle time of 18 μsec and a complete and versatile complement of peripherals. It has proved to be a surprisingly adequate instrument for a program of this complexity. The Breadboard programs and data are several times larger than the 16,000-word memory of the computer; to accommodate them, we have used magnetic tape peripheral memory in extensive and novel ways. Disc peripheral memory would, however, be essential for further development.

The principal routines and the syntactic/semantic rules of the DEACON Breadboard were programmed for the computer in LAP, an intermediary list-processing language. List-processing languages are a development of great importance in research having to do with advanced applications of computers, and almost all significant work in computational linguistics is based upon these languages. LAP was developed for the GE 225 as part of the DEACON project.[13]

The Breadboard grammar contains approximately 200 syntactic/semantic rules which are used in the analyses of sentences and preparation of programs to elicit required responses. A fully implemented system could be expected to be an order of magnitude larger in this respect.

The data base, together with those aspects of the vocabulary specific to the data base, was developed solely to facilitate the role of the Breadboard stated above. The data base utilizes a quasi-naval vocabulary and organization but the selection of this type of data was purely arbitrary. There are currently about 3000 items of data in the base, which could grow to several million items in a fully implemented system.

The Breadboard vocabulary consists of about 400 words of which approximately one-fourth are general or "functional" words such as "and," "what," and "of." This is a very high ratio of functional words to total vocabulary, as would be expected in a test vehicle. In a fully implemented system, a vocabulary of upwards of 40,000 words would be expected, of which some 1500 would be functional.

In summary, the DEACON Breadboard is clearly a test vehicle but, notwithstanding, it is a surprisingly powerful system which more than accomplishes its goals.

FUTURE DEACON DEVELOPMENT

To proceed from the Breadboard to a full-scale prototype is a task of considerable magnitude and one requiring some highly technical work. Three major aspects of this task are discussed in the following paragraphs.

First, the basic processing routines must be reprogrammed for a more adequate computer configuration. The peripheral memory must be discs instead of tapes. Remote typewriter input/output must be included in the system. Experience with the Breadboard has suggested modifications of internal formats that could cut processing time considerably. As for the principal routines themselves, the following revisions are anticipated. The sentence-parsing routine will be programmed to incorporate a variety of heuristics. At present, complexities of sentence structure may cause processing time in this routine to be totally unacceptable; but, through known techniques (some original to ourselves and some developed in other research groups) these can be brought under control. In addition, our experience indicates that an even closer tie between the syntactic and semantic aspects of sentence processing would be desirable. This implies a knitting together of two major processing routines.

Second, a much more complete set of syntactic/semantic rules must be developed. That is, we must develop an adequate grammar. There are grammars that are more complete than that used in Breadboard; certainly these grammar will be invaluable as we proceed, but they can by no means be directly or easily carried over to DEACON. An obvious difficulty is that the semantic categories used in DEACON (which are central to the semantic advance reflected by DEACON) are different from the categories used in other grammars. A comment or two may help to indicate the nature of the task of extending the current DEACON grammar. In the Breadboard grammar we do not make use of inflections;e.g., "ship" and "ships" are treated synonymously, as are "he," "him," and "they." The addition of distinctions between such words will alone increase the number of rules greatly; it will also add significantly to the grammatical power of the system and reduce processing time. Expanding the Breadboard vocabulary of general words from 100 to an anticipated 1500 will significantly increase the number of rules since many of these words will be directly involved in rule formation.

Third, certain revisions in the underlying structural aspects of the list language are strongly indicated by experience with the Breadboard. Moreover, it appears unlikely that further work will be done on the GE 225 computer. Consequently, an entire reworking of the underlying list-processing language is called for. It is possible that this could be carried out by suitable exploitation of the list-processing language IPL-V (which is available on a number of computers). In any event, there is a body of basic work that must be accomplished at this language level in order to exploit fully what we have learned from the Breadboard and to give a prototype system maximum grammatical effectiveness.

Resources and requirements for accomplishing the tasks discussed above include the following: First, the experience of the current team is a valuable resource. Second, this team must be augmented by senior people in several specialized areas. Specific requirements are for someone trained in modern structural linguistics and experienced in computational linguistics, and someone knowledgeable in the programming and operation of large computer systems using multiple peripherals and multiprocessing techniques. There is a very restricted population of both of these types of people. Third, direct and unconstrained access to an adequate computer is a necessity because further development will require sustained experimentation.

SECTION 2
THE CENTRAL CORE OF DEACON TECHNIQUES

THE SEMANTIC COUNTERPART OF SYNTAX

We are all familiar with the notion, often taught in elementary English classes, of diagramming, or parsing, a sentence. We know that prepositions are followed by noun phrases, transitive verbs require objects, etc. Modern structural linguists have extended our knowledge of the structure of language and provided new, more exact means of describing the grammar of English. One form in which much of English grammar can be described is in terms of phrase-structure rules. An adequate set of phrase-structure rules is in essence a precise set of rules by which sentences may be parsed. Since the syntactic rules of the DEACON Breadboard are phrase-structure rules, an example of such rules and their use in analyzing a sentence may be helpful at this point. Figure 1 is such an example.

The structural aspects of a sentence certainly contribute to the meaning of the sentence over and beyond the meanings of the individual words. For example, the following two sentences have nearly the same words but differ markedly in meaning:

> Beacon Hill was at its highest point in the golden age of Boston
> At the highest point in Boston was Beacon Hill golden in its age.

Thus the structure of the sentence carries semantic information. Somewhere within the structure of the sentence is the necessary information from which to program the implied relationships between the various words. If we can capture the nature of that information we can utilize it to instruct the computer how to interpret the significance of the sentence structure.

However, although it is the syntactic structure of the sentence that ties together the meanings of the individual words, the understanding of these semantic ties depends upon a prior understanding of how meaning is embodied in the words themselves. We first note that as far as meaning goes, certain words refer to external objects, classes of

Phrase structure rules:

1 article + noun phrase ← noun phrase

2 preposition + noun phrase ← prepositional phrase

3 noun phrase + prepositional phrase ← noun phrase

4 auxiliary verb + verb phrase ← verb phrase

5 verb phrase + prepositional phrase ← verb phrase

6 noun phrase + verb phrase ← sentence

Analysis, in linear form:

(((The ball)(in (the air))) ((will fall) (on (the ground))))
6 3 1   2 1   5 4   2 1

Analysis in tree form:

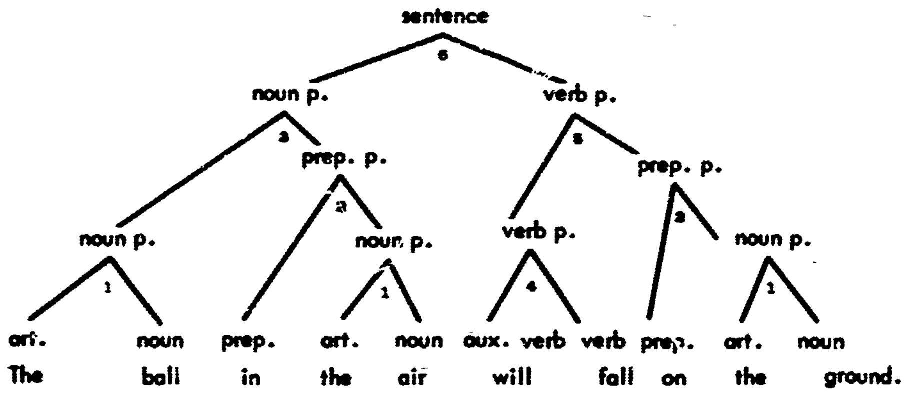


Figure 1. Examples of phrase-structure rules and the analysis of a sentence, in linear form and in tree form, through the use of these rules.

objects, and events, e.g., "Boston," "ships," "arrivals." Other words do not have such referents, e.g., "and," "what," "of." Therefore, we shall divide the vocabulary into two parts: referent words and functional words.

The sentence itself is characterized by three "constituents":

1. referent words
2. functional words
3. inflections, punctuation, and positional clues.

Classically, all words of the sentence are parsed using inflections and positional clues to determine the parsing, as in the above example. Our first significant step away from this classic process is to identify the functional words with the inflections and positional clues rather than with the referent words. Thus we parse the referent words, using functional words, inflections, and positional clues to determine the parsing. It is the referent words that have significant meanings, meanings which are interrelated by the sentence. And this interrelationship is established by the structure of the sentence using inflections, functional words, position, and punctuation.

Let us look more closely at referent words. Take "Boston," for example. Clearly it refers to a great city, to its topography, economic and political geography, history, relationships with other entities, etc. Some aspects of Boston are indeed "real" in the sense that we can see them, such as Beacon Hill and Durgen Park. Certainly some aspects exist only in our own memory. Some aspects you may know; other aspects are not known to you, possibly not to anyone. What we wish to call attention to here is that Boston to you can mean no less and no more than what is in your memory about Boston. Whatever it may be in reality, the word "Boston" can for you refer only to what you know about Boston. This idea is even more applicable to the computer. Whatever the computer is to understand by "Boston" must already have been placed in its memory in a way appropriate for its retrieval upon reference to that word.

If we were to look into the computer at any given point in time and follow the links that lead from some input register containing the word "Boston," we should find a body of material which can be called the computer's record concerning Boston. Further, this record concerning Boston is structured in some convenient manner. Possibly, for example, it is the region from memory location 1000 to memory location 1500; perhaps the first 40 locations give the longitude and latitude of points on its periphery, the next 200 the name and total yearly value added for the 100 principal industries.

It would be convenient to have the various records corresponding to referent words classified into a few types. For example, the records corresponding to names might all be in the form of lists with appropriate sublists (so called "list structures"); adjectives, whose reference is a scattering of keys which identify those things having a given property, would always be attached to the list structures of the corresponding things in a specific structural way; and the referents of verbs might be stored as subroutines. What we are suggesting here is that the part of speech assigned to a word identifies the way in which the referent of that word is stored in memory. This idea is not entirely new; it is inherent in the work on semantic categories of Dr. A. Tarski[8] and Dr. S. Lesniewski. In their work on the formal languages of symbolic logic, "parts of speech" entirely different from those of classical linguistics were used. Nevertheless, the rules of grammar for these languages otherwise resemble the rules linguists use. The distinctive difference, from a fundamental point of view, was that the part of speech not only identified the building blocks for syntactic constructions, but they also identified the structural aspects of the meanings of referent words to that degree necessary to establish the meaning of the entire sentence. Specifically, both the rules of formation and the rules of truth were given in terms of the parts of speech.

It is not at all clear that the parts of speech of classical linguistics carry such structural implications. Therefore, it has been necessary for us to find alternative parts of speech that would be adequate both for the formulation of the rules of grammar and for the identification of the structures in memory of the references for referent words. The new family of parts of speech has been found to bear more than a superficial resemblance to classical parts of speech.

Before turning directly to an enumeration and discussion of the parts of speech used in the DEACON Breadboard, let us evaluate the position in which we now find ourselves. What we are looking for is a means of grasping and utilizing the semantic information that is contained in the structural or syntactic aspects of a sentence. We now understand that the structure and functional words of a sentence establish appropriate relationships between its referent words. The syntactic aspects of these relationships are given in terms of phrase-structure rules which establish how compound phrases can be built from constituent words or phrases having the parts of speech specified by the rule. We are also stipulating that the parts of speech to be used not only function in conjunction with the syntactic rules, but also carry implications of the structures of the material in memory referred to by the corresponding words.

It remains to take the key step in the whole process. We start with an example. Consider

1. the two words "red" and "ships,"
2. the phrase structure rule "adjective + noun ⇒ noun phrase."

In the first place, the rule clearly applies to the two words to form the phrase "red ships." Suppose for the moment we think of "noun" as indicating an item or list of items, each of which is accompanied by the values of its attributes; thus "ships," being a noun, is construed as referring to the list of all ships, showing for each its location, color, and destination as follows:

ships	location	color	destination
ship$_1$	Boston	white	New York
ship$_2$	Norfolk	red	Boston
ship$_3$	San Diego	gray	⋮
ship$_4$	Boston	red	
⋮	⋮	⋮	

Further, we think of "adjective" as indicating the specified value of attributes attached to and characterizing the various items in memory; thus "red," being an adjective, is construed as being attached to the attribute "color" for those and only those items which are colored red. What now does the compound phrase "red ships" refer to? Since it is a noun phrase, it must refer to a list; obviously, the list of red ships, i.e., the sublist of the ship list consisting of those and only those ships whose value for the attribute color is red, as indicated below:

red ships	location	color	destination
ship$_2$	Norfolk	red	Boston
ship$_4$	Boston	red	⋮
⋮	⋮	⋮	

This same analysis of the phrase "red ships" can be applied to any adjective + noun phrase:

- The phrase "adjective + noun" refers to the sublist of the list named by the noun consisting of those and only those things for which the appropriate attribute has the value indicated by the adjective.

The semantic rule is clearly the exact counterpart of the syntactic (phrase-structure) rule "adjective + noun ⇒ noun phrase." Indeed they are two aspects of the same linguistic rule, one giving the conditions under which a new phrase can be constructed, the other establishing what the referent of the resulting phrase is to be.

Having established the structural significance of the new parts of speech, it is possible to find the semantic counterpart of each rule of grammar. The resulting syntactic/semantic rules then allow for a complete analysis of a sentence. The syntactic aspects of the rules are used to parse the sentence, showing how referent words are compounded into phrases and these in turn into more complex phrase constructions. In turn, the semantic aspects show how to manufacture the referents for its constituents, until the referent for the entire sentence stands revealed. For a declarative sentence, this will be its truth or falsity; for an interrogative sentence, this will be the answer; and for an imperative sentence, the course of action.

PARTS OF SPEECH AND RULES OF GRAMMAR

We now return to the discussion of our new parts of speech, for it is in terms of these that our syntactic/semantic rules will be given. Once this has been done, the overall method can be clearly shown in a variety of examples.

The parts of speech used in DEACON are closely related to the underlying list-processing language. The first requirement in selecting a family of parts of speech is generality; we do not want a separate part of speech for each minor modification in memory format. The second requirement is universality; the structural categories we do pick must be such as to encompass all memory formats which would be desirable in the kinds of applications anticipated. Third, the proliferation of rules, on the one hand, and internal ambiguities, on the other, must be minimized in a balanced way. Suffice it to say here that these conditions impose difficulties of selection if approached de novo. It comes down to a fundamental question of all thought; in what ways should we structure our knowledge of our world so that we can deal with it efficiently? Certainly one answer is: in that way we have found, through experience, to be efficient. From this point of view, the categories of list processing suggested themselves.[11]

This selection establishes biases in the efficiency of the systems built upon it. Where the universe of discourse to be dealt with by the system is structured in a natural way in list-structural form, the system will be efficient. However, it is easy to imagine areas of application where a system based upon our categories would be inefficient, for example, a subject area where spatial layout is important. With our parts of speech, communications using the range of prepositions: "on," "under," "over," "beside," would be very difficult to accommodate. In summary, (1) the selection of parts of speech has a fundamental effect on the balance of efficiencies of the resulting system, and (2) the selection made for the Breadboard, although thoughtfully made, is in no sense ultimate. It is precisely this area of the choice of parts of speech that we believe to be the most fruitful domain of further research.

Before presenting the parts of speech we are using, a few descriptive remarks concerning list processing may be helpful. In the terminology of list processing, a list structure L is:

1. a list or sequence S_1, $S_2 \ldots S_n$ of items such that each item S_i may be either itself a list structure or may be simply a number or name.

2. a list of pairs $\langle a_1, v_1 \rangle$, $\langle a_2, v_2 \rangle \ldots \langle a_m, v_m \rangle$, called the description list of L, where each pair $\langle a_i, v_i \rangle$ is thought of as an attribute a_i of L together with its value v_i for L .

In the example in Figure 2, the Task Force 60 as a list consists of the Task Group 60.1 and the Forrestal. As a list structure it is more extensive: it has a description list giving three attribute-value pairs (Commander: Jones, Flagship: Forrestal, and Operational superior: 6th Fleet), as well as two sublists (Task Group 60.1 and Forrestal), and two sub-sublists (King and Long).

The parts of speech adopted for the DEACON Breadboard are:

L:	list
D:	entity
A:	attribute
V:	value
R:	routine
N:	number
X:	variable
T:	truth value
F:	functional word.

07507

```
Task Force 60
                    Commander: Capt. Jones
                    Flagship: Forrestal
                    Operational superior: 6th fleet

       Task Group 60.1
                    Commander: Capt. Smith
                    Flagship: King

            King
                    Commander: Cmdr. Thomas
                    Location: Boston

            Long
                    Commander: Cmdr. Tracy
                    Location: long. 60 E, lat. 80 N

       Forrestal
                    Commander: Capt. Murray
                    Location: Boston
```

Figure 2. Example of a list structure.

The first two, L and D, should be considered together. Each stands in its appropriate sense for a list structure. Consider the list structure of the example: Task Force 60. As a list consisting of Task Group 60.1 and the Forrestal, it has the part of speech L . Thus it is used as an L in the sentence: "Is the Forrestal in Task Force 60?" As an entity, or an object of interest in its totality, it has the part of speech D . Thus it is used as a D in the sentence: "Capt. Jones is commander of Task Force 60." It is usually the case that if a word has part of speech D or L, it has both parts of speech. (The assignment of multiple parts of speech is, of course, not uncommon in classical linguistics.) A counter example might be the list structure King in the example, since it would most likely be assigned only the part of speech D .

Attribute A, and value V, of course, refer to words which occur on description lists. Thus in the example in Figure 2, "Flagship" has part of speech A, and "King" part of speech V . (King also is a D, as indicated above.)

Some words refer to computer subprograms, or routines, and are given the part of speech R . There is a high correlation between our part of speech R and the classical part of speech "verb." However, certain other types of words such as "maximum" also may be R's .

The N (Number) category represents a pure numeric, such as 571 or 86.

Variable X plays the role of pronoun. Thus words like "he" and "it" are assigned the part of speech X .

Truth value T is the part of speech assigned to complete declarative sentences or clauses. A sentence, following the practice in symbolic logic going back to Frege, refers to either truth or falsity, and it is this convention that is represented by the part of speech T .

Functional words play a role distinctly different in the DEACON system than do referent words. The part of speech F represents a word which shows structural relationships between and among other words in a sentence. Functional words include the conventionally accepted conjunctions, prepositions and articles, and a few other miscellaneous words. In effect, each functional word (or, more precisely, each set of synonymous functional words) constitutes a separate part of speech, since each must be used specifically in forming rules of grammar. That is, a rule using only the category F for functional words would not be adequate to reflect obviously required distinctions.

Here is an example of assignment of parts of speech:

F	A	F	D	R	V	D
The	commander	of	$\overbrace{\text{Task Force 61}}$	is	Adm.	Black.

It is in terms of these parts of speech that the syntactic/semantic rules are formulated. A list of the rules used in the DEACON Breadboard is in the appendix. Here we shall illustrate the rules and techniques by means of examples.

Consider the sentence, "The commander of Task Force 61 is Adm. Black." The following rules are applied in its analysis.

Rule 33: $A \frown \text{of} \frown D \Rightarrow V$ where the resulting V is the value of the attribute A for the entity D .

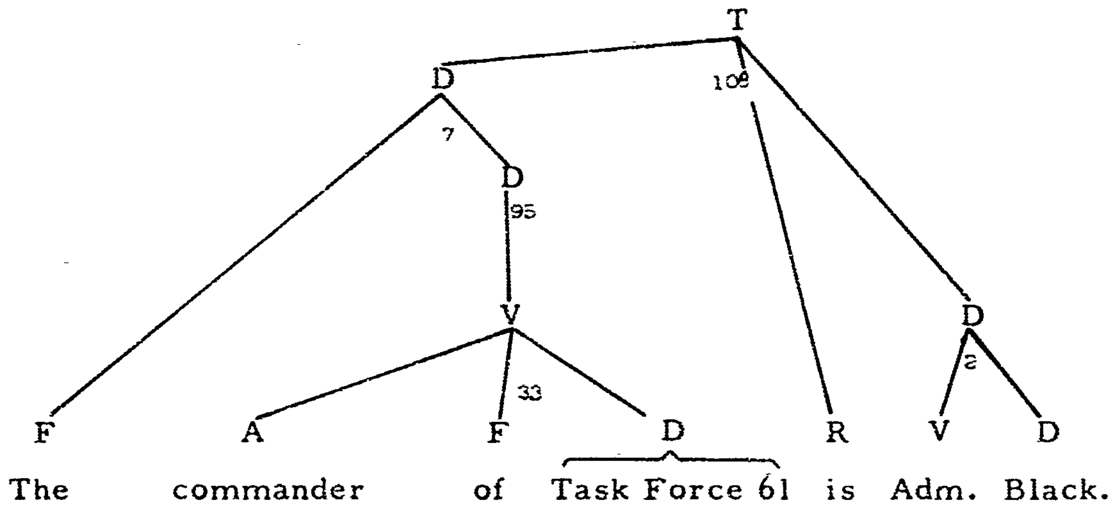
Rule 2: $V^\frown D \Rightarrow D$ where the resulting D is the same as the constituent D if the constituent has the value V for the appropriate attribute.

Rule 95: $V \Rightarrow D$ this rule applies only when a value also is an entity in its own right. Thus Forrestal may be the location of an officer and also a ship to be considered as an entity.

Rule 7: $\text{the}^\frown D \Rightarrow D$ "the" has an important function as discussed below. Here we shall consider 'the D" as synonymous with the D itself.

Rule 108: $D^\frown R^\frown D \Rightarrow T$ this rule may be considered as the "transitive verb" rule in the sense that the routine R requires two D's as arguments. The result is an indication of either "true" or "false."

The parsing of the sentence is illustrated below (the number at each node represents the rule being applied, and the letter at each node represents the part of speech of the phrase that results from applying the rule):

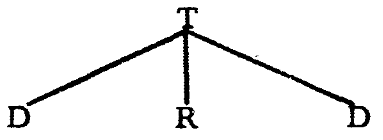


The semantic transformations corresponding to each rule are now applied, starting at the bottom of the diagram. Thus

- "Adm. Black" is replaced by "Black" since he is indeed an admiral according to the data in memory and therefore meets the requirements of Rule 2.

- the Task Force 61 record is checked for the attribute commander and its value, Black, is located and used to replace "commander of Task Force 61." Although this "Black" has part of speech V, it is converted to part of speech D by applying Rule 95.
- "the Black" is replaced by "Black," through application of Rule 7.

We now have:



Black is Black

Going to the data for the routine corresponding to "is" yields that "D_1 is D_2" is true if D_1 is D_2 . Therefore the sentence as a whole is true.

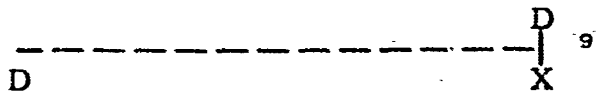
SOME SPECIAL CLASSES OF RULES

Not all of the syntactic/semantic rules used in the Breadboard are as straightforward as those illustrated in the above example. Certain classes of rules will be discussed here, readying us for a wider range of examples.

First consider the question of pronouns, that is, words whose part of speech is X . The principal pronoun rule is:

Rule 9: D . . . X ⇒ D . . . D

This rule, in the special vocabulary of phrase-structure grammar, is said to be discontinuous and context dependent. The ellipsis indicates that the constituents D and X do not have to be contiguous in the sentence, hence "discontinuous." In applying the rule, the D constituent does not change, whereas the X constituent is replaced by the D . The D constituent forms a context in which the X can be successfully developed, but a context which itself is not modified; hence a "context dependent" rule. An example of its application is:



Black is an admiral and he is on the Forrestal.

Applying Rule 9 replaces "he" with "Black," yielding:

Black is an admiral and Black is on the Forrestal.

Pronouns have one other important feature—namely, each carries with it a list L called its range. For example, for the data base of the Breadboard, the range of "he" is the list of personnel, since we have no women mentioned therein. Incidentally, the range of "she" is the ship list. In the semantic part of Rule 9, a check is made to ensure that the D constituent is in the range of the X constituent before the rule is allowed to apply. Thus in the example:

"Forrestal and Black will arrive in Boston and he will arrive today,"

"he" is unambiguous since "Black" is the only preceding D that is in its range. Of course, the following example is ambiguous:

"Black and Jones will arrive in Boston and he will arrive today."

Consider next the word "the." In the example:

"Forrestal and Black will arrive in Boston and the ship will arrive today."

One cannot consider "ship" as a D since what ship is unknown. The phrase "the ship," however, begs for an antecedent, just as the pronoun "he" did in the example several lines above. Therefore we have the rule:

Rule 8: $\text{the}^\frown L \Rightarrow X$

where the range of the resulting X is the list L . In our example, "the ship" becomes a pronoun with the ship list as its range. After applying Rule 8, we can then apply Rule 9, getting:

"Forrestal and Black will arrive in Boston and Forrestal will arrive today."

An important class of rules has to do with quantifiers and generators. The quantifier is an important notion taken over from symbolic logic. It refers to such words as "all," "any," "some," and, as we shall see, others as well. Thus "all ships" and "some ships" are quantified forms of the ship list. The generator, on the other hand, is a notion from list processing. A generator concerns a list L, a process P, and an output Q . The generator presents to the process P each element of the list L in turn. After the process P is applied to an element, the output is modified accordingly and a decision is made whether or not to continue. The generator is thus completed, with the resultant the final Q, either when all elements of L are generated or at some intermediate point at which the conditions of generation are satisfied.

Quantifiers and generators are intimately related.* Indeed, in the DEACON Breadboard, the linguistic objects, "quantifiers," correspond to a certain class of semantic objects, "generators," whose process P is the semantic analysis of a sentence or clause resulting from the replacement of the quantified list by the generated element. This relation may be clarified by an example. Consider the sentence:

All ships are in port.

The quantified expression "all ships" indicates that a generator should be set up generating the elements of the ship list. As each element, say ship_i, is generated, the resulting clause

ship_i is in port

is considered. Upon analysis, it is found to be either true or false. If true, the generator proceeds to the next element, ship_{i+1}, and so on. If false, the generation operation is discontinued and the output set false. If the whole list of ships is generated, the generation operation is completed and the output set true. In short, "All ships are in port" is true if, for every ship_i, ship_i is in port; and is false if, for some ship_i, ship_i is not in port.

Similarly in the sentence:

Some ships are in port

the quantified expression "some ships" sets up a generator of the ship list. The "some" generator stops with output true upon the first clause "ship_i is in port" found to be true. If all elements are generated, the "some" quantifier is completed with output set false.

In the sentence:

What ships are in port?

*This relationship was pointed out to us by Dr. Robert Lindsay.

Applying Rule 9 replaces "he" with "Black," yielding:

> Black is an admiral and Black is on the Forrestal.

Pronouns have one other important feature—namely, each carries with it a list L called its range. For example, for the data base of the Breadboard, the range of "he" is the list of personnel, since we have no women mentioned therein. Incidentally, the range of "she" is the ship list. In the semantic part of Rule 9, a check is made to ensure that the D constituent is in the range of the X constituent before the rule is allowed to apply. Thus in the example:

> "Forrestal and Black will arrive in Boston and he will arrive today,"

"he" is unambiguous since "Black" is the only preceding D that is in its range. Of course, the following example is ambiguous:

> "Black and Jones will arrive in Boston and he will arrive today."

Consider next the word "the." In the example:

> "Forrestal and Black will arrive in Boston and the ship will arrive today."

One cannot consider "ship" as a D since what ship is unknown. The phrase "the ship," however, begs for an antecedent, just as the pronoun "he" did in the example several lines above. Therefore we have the rule:

Rule 8: $\text{the}^\frown L \Rightarrow X$

where the range of the resulting X is the list L . In our example, "the ship" becomes a pronoun with the ship list as its range. After applying Rule 8, we can then apply Rule 9, getting:

> "Forrestal and Black will arrive in Boston and Forrestal will arrive today."

An important class of rules has to do with quantifiers and generators. The quantifier is an important notion taken over from symbolic logic. It refers to such words as "all," "any," "some," and, as we shall see, others as well. Thus "all ships" and "some ships" are quantified forms of the ship list. The generator, on the other hand, is a notion from list processing. A generator concerns a list L, a process P, and an output Q . The generator presents to the process P each element of the list L in turn. After the process P is applied to an element, the output is modified accordingly and a decision is made whether or not to continue. The generator is thus completed, with the resultant the final Q, either when all elements of L are generated or at some intermediate point at which the conditions of generation are satisfied.

Quantifiers and generators are intimately related.* Indeed, in the DEACON Breadboard, the linguistic objects, "quantifiers," correspond to a certain class of semantic objects, "generators," whose process P is the semantic analysis of a sentence or clause resulting from the replacement of the quantified list b [illegible] generated element. This relation may be clarified by an exampl [illegible] er the sentence:

All ships are in port.

The quantified expression "all ships" indicates that a generator should be set up generating the elements of the ship list. As each element, say ship$_i$, is generated, the resulting clause

ship$_i$ is in port

is considered. Upon analysis, it is found to be either true or false. If true, the generator proceeds to the next element, ship$_{i+1}$, and so on. If false, the generation operation is discontinued and the output set false. If the whole list of ships is generated, the generation operation is completed and the output set true. In short, "All ships are in port" is true if, for every ship$_i$, ship$_i$ is in port; and is false if, for some ship$_i$, ship$_i$ is not in port.

Similarly in the sentence:

Some ships are in port

the quantified expression "some ships" sets up a generator of the ship list. The "some" generator stops with output true upon the first clause "ship$_i$ is in port" found to be true. If all elements are generated, the "some" quantifier is completed with output set false.

In the sentence:

What ships are in port?

*This relationship was pointed out to us by Dr. Robert Lindsay.

the quantified expression sets up a generator. The entire ship list is generated and each ship_i for which "ship_i is in port" is true is put on the output list Q . Thus the output Q is precisely the list of those ships which are in port. Similarly in "How many ships are in port?" the output from the "how many" generator is the number of ships which are in port.

A final note tying quantifier/generators to pronouns. If no antecedent is found for a pronoun X, its range L is automatically generated as if the pronoun were replaced by "all L ." Consider the following:

Are the personnel of CVA 59 in port?

Applying Rule 35 ($L \frown \text{of} \frown L \Rightarrow L$), the list L' of personnel of CVA 59 is obtained. Rule 8 ($\text{the} \frown L \Rightarrow X$) then makes a pronoun out of "the L' ." This is ultimately replaced by "all L' " and generated by Rule 189 ($X \Rightarrow D$), the resultant D indicating the part of speech of the element generated. At this point, the sentence has been reduced to:

"Is (personnel_i of CVA 59) in port?"

From this point, it is clear that the result of the "all" generator will yield the desired result.

The final aspect of grammar to be discussed in this summary document is the method of handling verbs, or more precisely R's . As a prototype of R-words, we shall study the verb "to arrive." Our point of view is that "arrive" refers to a class of events, namely the class of all arrivals. More precisely, it is the class of all recorded or scheduled arrivals in the data available to the computer.

The routine referred to by "arrive" has one argument, that is to say, it is an intransitive verb taking a subject. It also has a tense. Consider for a moment a two-dimensional coordinate system where tense is represented along the horizontal axis and all possible subjects are spread along the vertical axis, as shown in Figure 3. A point in this system with coordinates $\langle t_0, s_0 \rangle$ is thus a potential arrival, the arrival of s_0 at time t_0 . If it is recorded in the data base that s_0 did arrive at t_0, then this point is in the class of events named by "arrive." This class of events is represented graphically by a scattering of points in the plane.

The sentence: "The Forrestal will arrive tomorrow," is true if and only if the point A is in the class named by "arrival," i.e., it is a

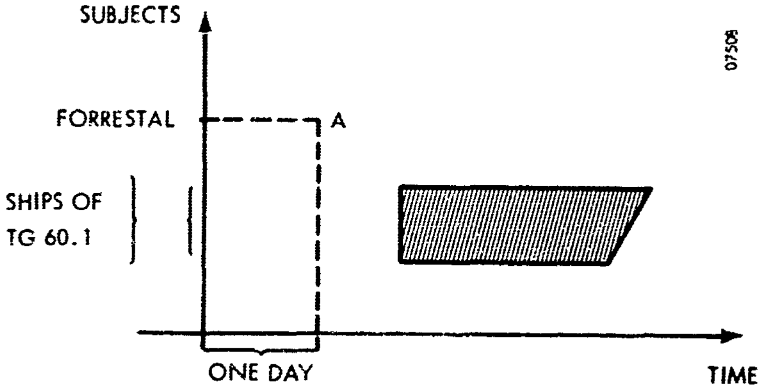


Figure 3. Characterization of events in terms of subject involved and time of occurrence.

bona fide scheduled arrival. The question "Will any ships in TG 60.1 arrive after two days from now?" should bc answered true if and only if there is at least one point in the shaded area that also is a point of the class named by "arrival." We thus see that the routine referred to by the R-word "arrive" can be construed as a search routine that searches an area of the time-subject plane delineated by the tense and subject of the sentence. The precise programming of the routine will depend on how the records on arrivals are put into memory.

In the DEACON Breadboard, the tense of an R-phrase is defined in terms of two points: "now" and a time reference. Rather than giving an exhaustive treatment here, we will illustrate the notion by example. Thus we have the future tenses indicated in Figure 4.

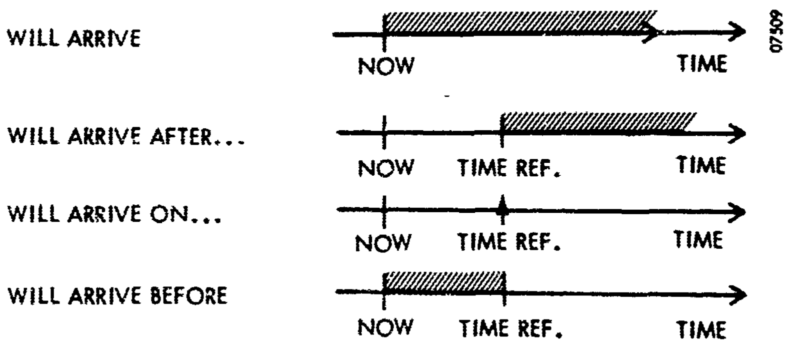


Figure 4. Characterization of future tenses of verbs.

Thus the tense of the R-word is modified by adverbial modifiers of tense and auxiliaries. This is handled by rules such as:

Rule 46: will . . . R ⇒ R

Rule 66: R . . . after N ⇒ R

where the tense character of the initial R is modified and, in the case of Rule 66, a time reference is added, to obtain the resulting R .

Treatment of adverbial modifiers of the R-phrase which are not related to tense will be illustrated by the sentence:

"The Forrestal will arrive in Boston tomorrow."

So far we have thought of "arrive" as naming a subset of the time-subject plane. The phrase "in Boston" adds a new dimension to this phase space for arrive, as shown in Figure 5, but at the same time delimits the area of search in this space.

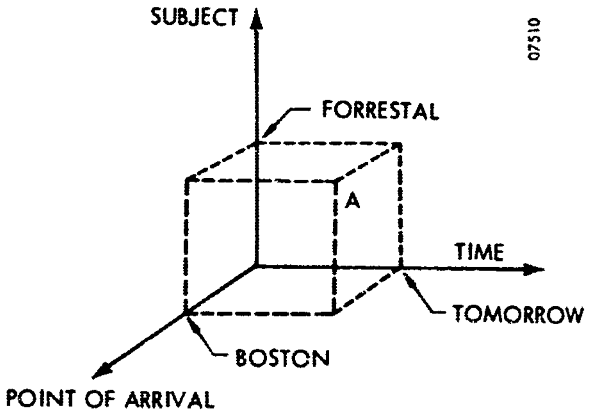


Figure 5. Characterization of events in terms of subject involved and time and place of occurrence.

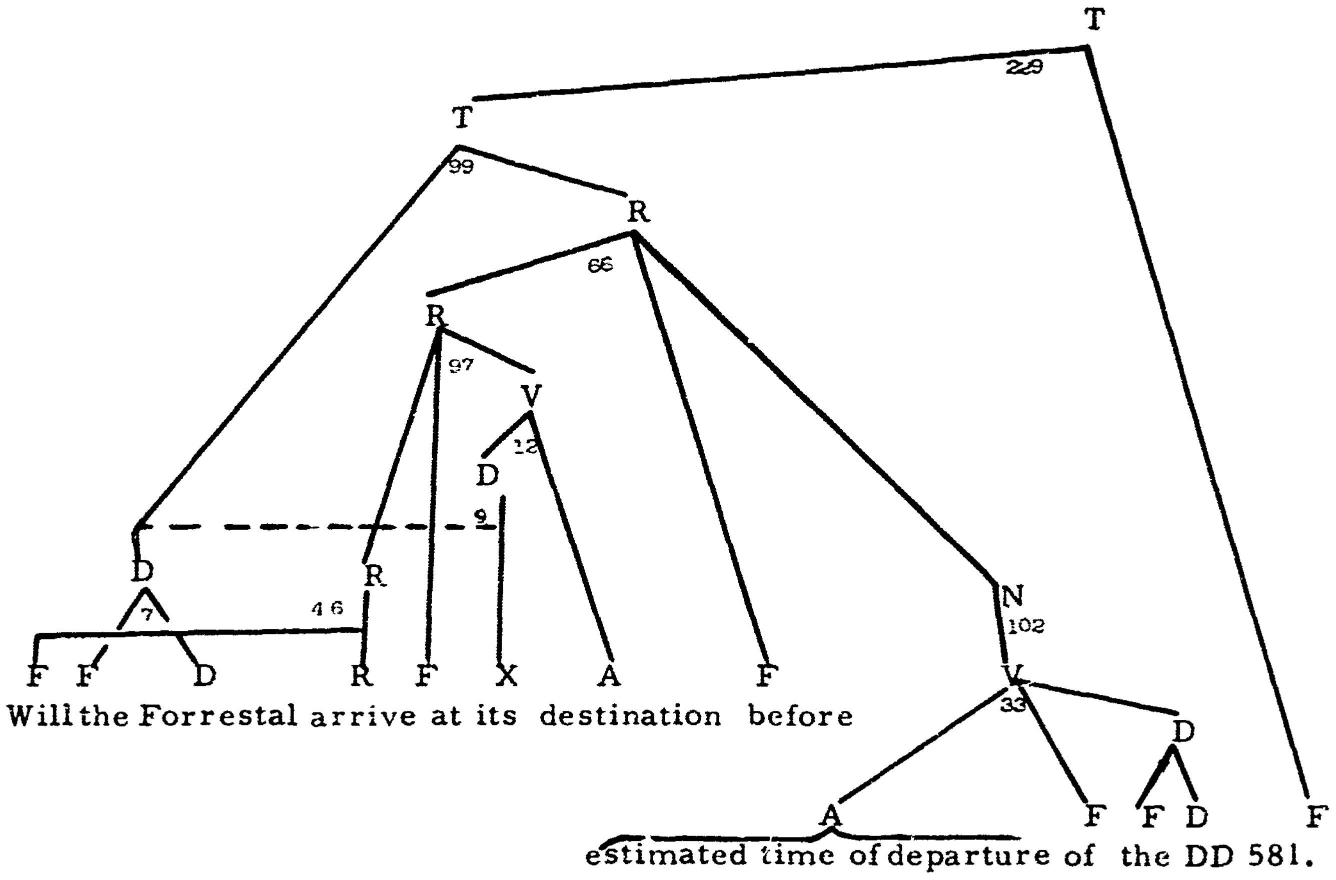
Thus the given sentence is true if and only if the point A is in the set of points named by "arrive." The sentence: "A ship in Task Force 60 will arrive after tomorrow in an East Coast port," establishes a region of this three dimensional space that is now to be searched for the existence of a point of the arrival set. The "arrive" routine carries out this search. The arrive routine does not have to anticipate the adverbial modifier since the modifying clause is constructed to carry the necessary information to allow proper augmentation of the search routine in that area of the data base that the modifier keys. Thus the search routine is a generalized routine with subject and tense, or subject, object, and tense, as a minimal base for search, augmented by those dimensions of search keyed by modifiers. The addition of modifiers is carried out by such rules as:

Rule 105: $R \ldots \mathrm{in} \frown V \Rightarrow R$
Rule 79: $R \ldots \mathrm{between} \frown N \frown \mathrm{and} \frown N \Rightarrow R$
} where that part of the R-word record that establishes the search area is appropriately augmented.

EXAMPLES

In this paragraph we shall simply present several examples to illustrate the notions discussed above. (Numbers indicate rules applied.)

Sentence: Will the Forrestal arrive at its destination before estimated time of departure of the DD 581 ?


T
229
T
99
R
66
R
97
V
12
D
9
D
R
7
46
N
102
F F D R F X A F
Will the Forrestal arrive at its destination before
V
33
D
A F F D F
estimated time of departure of the DD 581.

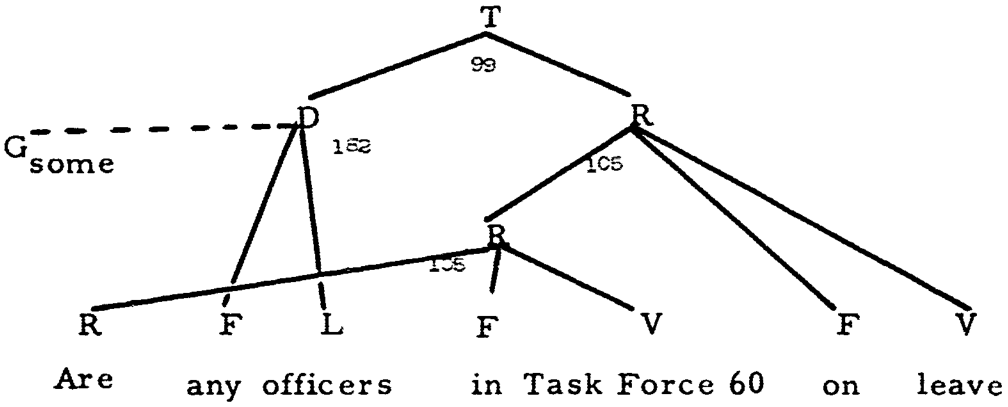
"Its" becomes "Forrestal"
"Forrestal destination" becomes "Norfolk"
"estimated time of departure of DD 581" becomes "7/20/63."

Incorporating the appropriate verb modifiers, the sentence is thereby reduced to:

$$\text{"Forrestal} \left\{ \text{arrives} \begin{array}{l} \text{before 7/20/63} \\ \text{at Norfolk} \end{array} \right\} \text{."}$$

The search routine finds in the Forrestal record the entries showing arrival at Norfolk on 8/31/63, and thus the sentence is marked false.

Sentence: Are any officers in Task Force 60 on leave?



The interesting aspect of this example, other than the "some" generator, is the escalation necessary to accomplish the "in Task Force 60." Task Force 60 is a list, but not of officers. However, Task Force 60 is a value of the attribute operational superior. After a generation of an element of the officer list, say Smith, we must check to see if Smith is in Task Force 60. As a matter of fact, the following relations are in data:

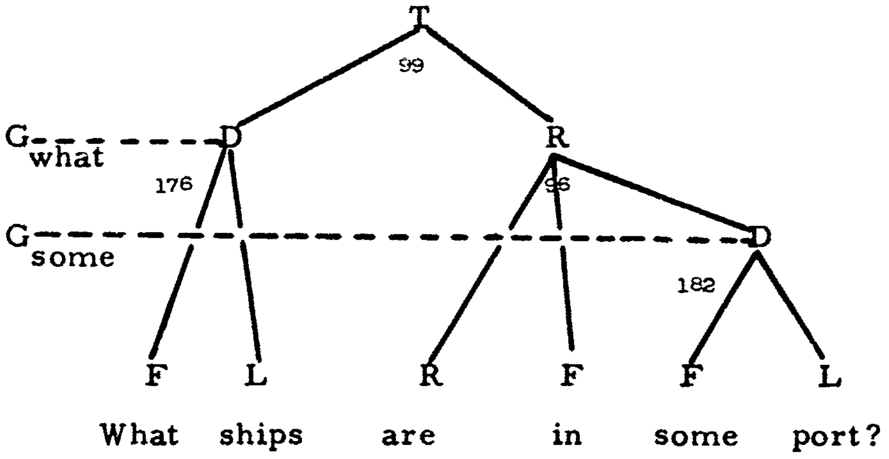
operational superior of Smith is MMA 15

operational superior of MMA 15 is TG 60.2

operational superior of TG 60.2 is TF 60.

This law of escalation has been found to be applicable in general and is built into the semantic aspects of the appropriate routines. It is found that Jones is in TF 60 and is on leave, so the sentence is true.

Sentence: What ships are in some port?



The what-generator first presents a ship$_i$. Then the some-generator generates the port list. If ship$_i$ is in fact in some port, say port$_j$, the some generator gives the output "true," which in turn signals the what-generator to add ship$_i$ to its output list.

These three examples illustrate the techniques of syntactic/semantic analysis used in the DEACON Breadboard.

SECTION 3
ORGANIZATION OF THE DEACON BREADBOARD

PRINCIPAL INPUTS

The principal inputs to the DEACON Breadboard are the data inputs, including the sentences to be processed, and the principal routines.

Data Inputs

Data inputs consist of the following:

1. Vocabulary: Each vocabulary word is followed by its associated definitions. Each definition consists of (a) part of speech, (b) certain related word information, e.g., tense for R-words, and (c) address in memory of the corresponding data, if applicable, e.g., the address of the corresponding list in data for L-words.

2. Syntactic/semantic rules: Each rule has three parts: (a) the sequence of constituent phrases, e.g., for Rule 33, $A^\frown of^\frown D \Rightarrow_{\circledast} V$, this would be simply "$A^\frown of^\frown D$"; (b) the routine that applies the rule whenever applicable, building the resulting compound phrase, etc; (c) the semantic transformation which operates on data to produce the new referent for the resulting phrase.

3. Data: The only restriction on the data is that it be in list processing formats.

4. Sentences to be processed.

Principal Routines

The principal routines are the following:

1. DICTPR (Dictionary Preparation): It processes the vocabulary and builds the internal dictionary.

2. RULEPR (Rules Preparation): It processes the rules list into a format more convenient for internal processing.

3. SENTPR (Sentence Preparation): It reads the next sentence to be processed from the card reader and formats it for internal processing.

4. ANALPR (Analysis Preparation): It prepares the initial "subgoals," i.e., internal, unambiguous forms of the sentence, prior to syntactic analysis.

5. REGIME: It carries out the syntactic analysis.

6. TRAN (Application of the Transformations): It carries out the semantic analysis.

7. ANOUT (Answer Out): It prepares and outputs the answer.

8. START: The executive routine for final processing.

These 12 packages are the major parts of the DEACON Breadboard. A description of what each does as part of the whole will now be given in two parts: first, the preparatory stage; second, the processing stage.

PREPARATORY STAGE

The preparatory stage precedes the reading in of the query or instruction sentences. The result is the building of three tapes and loading of the executive START routine. These tapes are referred to as Tapes 1, 2, and 3 in Table 1. Much of this preparation is simply copying principal routines from the DEACON library tape onto Tape 1. Therefore, only those tape entries involving processing will be discussed.

Processing The Vocabulary

There are both external words and internal words. The external words are the English words which are used to communicate with the computer. These external words may, on the one hand, have several definitions (e.g., there may be both a ship King and an officer King; one word "Forrestal" may be both a D and a V) and, on the other hand, there may be several external words having the same definitions (e.g., "Forrestal" and "CVA 59" are synonymous). Further, a sequence of several external words may function as an idiom and, thus, have a single definition. The internal words are in one-to-one correspondence with the definition; indeed they may be considered as names of the definitions. Thus, the internal words are uniquely defined.

Table 1. Status of Tapes 1, 2, and 3 at beginning of processing.

Tape 1	Tape 2	Tape 3
SENTPR	Definition 1	Spelling of first word
ANALPR	Definition 2	Spelling of second word
Dictionary 1	Definition 3	Spelling of third word
Dictionary 2	⋮	⋮
⋮	(End of file)	(End of file)
(End of file)		
REGIME		
Rules		
semantic transformation for Rule 1		
semantic transformation 2		
semantic transformation 3		
⋮		
(End of file)		
TRAN		
Data		
ANOUT		

The input vocabulary states definitions for each external word. First these are gathered and each distinct definition assigned a name, i.e., the internal words. At this point the input vocabulary gives for each external word the internal words that correspond to it. The definitions are then put out on Tape 2 as individual records, each record indexed by its internal word name. The spelling of the external words are put out on Tape 3 as individual records, each indexed by all internal names to which it corresponds. Finally one or more dictionaries are built. It is in these dictionaries that the external words are looked up, the entry for each external word being the corresponding internal words.

The dictionary is built in a tree pattern, which may best be clarified by an illustration. Suppose we wish a dictionary with the words "new," "next," "an," "a," "ant," and the idiom "New York." Suppose also that "new" has two definitions, while the rest have only one. The organization is shown in Figure 6. When put into list processing formats, the lists in the diagram labeled "letters" are description lists with the letters occurring in them as the attributes. Thus, to look up, say, "new":

1. find the value L_N of the attribute N in the dictionary list
2. find the value L_{NE} of the attribute E in the L_N list
3. find the value L_{NEW} of the attribute W in the L_{NE} list
4. since the letters of the word are exhausted, the entries in the list L_{NEW} are the corresponding internal words.

We see by the idiom mark in the L_{NEW} list, that "new" is also the first word of an idiom. If a word follows "new" in the sentence we are working on, the idiom-marker list replaces the dictionary list in looking up the next word.

Processing The Rules

The third part of each rule, namely, the semantic transformation part, is put out on Tape 1, each as a separate record indexed by the rule number. The remainder of the rules are arranged for convenient internal processing.

Preparation of the Query and Instruction Sentences

The GE 225 does not have an input console typewriter. Therefore, input is through the card reader. Query sentences are key-punched into standard punched cards in normal English sentence format. As many cards as needed may be used for a sentence. Several sentences may be stacked together, with a marker card between the last card of one sentence and the first card of the following. Because of the desirability for extensive output for debugging purposes, all output is through the high-speed printer.

PROCESSING STAGE

Step 1

Tape 1 is rewound and SENTPR is read in and executed. SENTPR checks the card reader and, if a sentence is waiting, reads it into memory. It then puts the sentence into a format convenient for further processing.

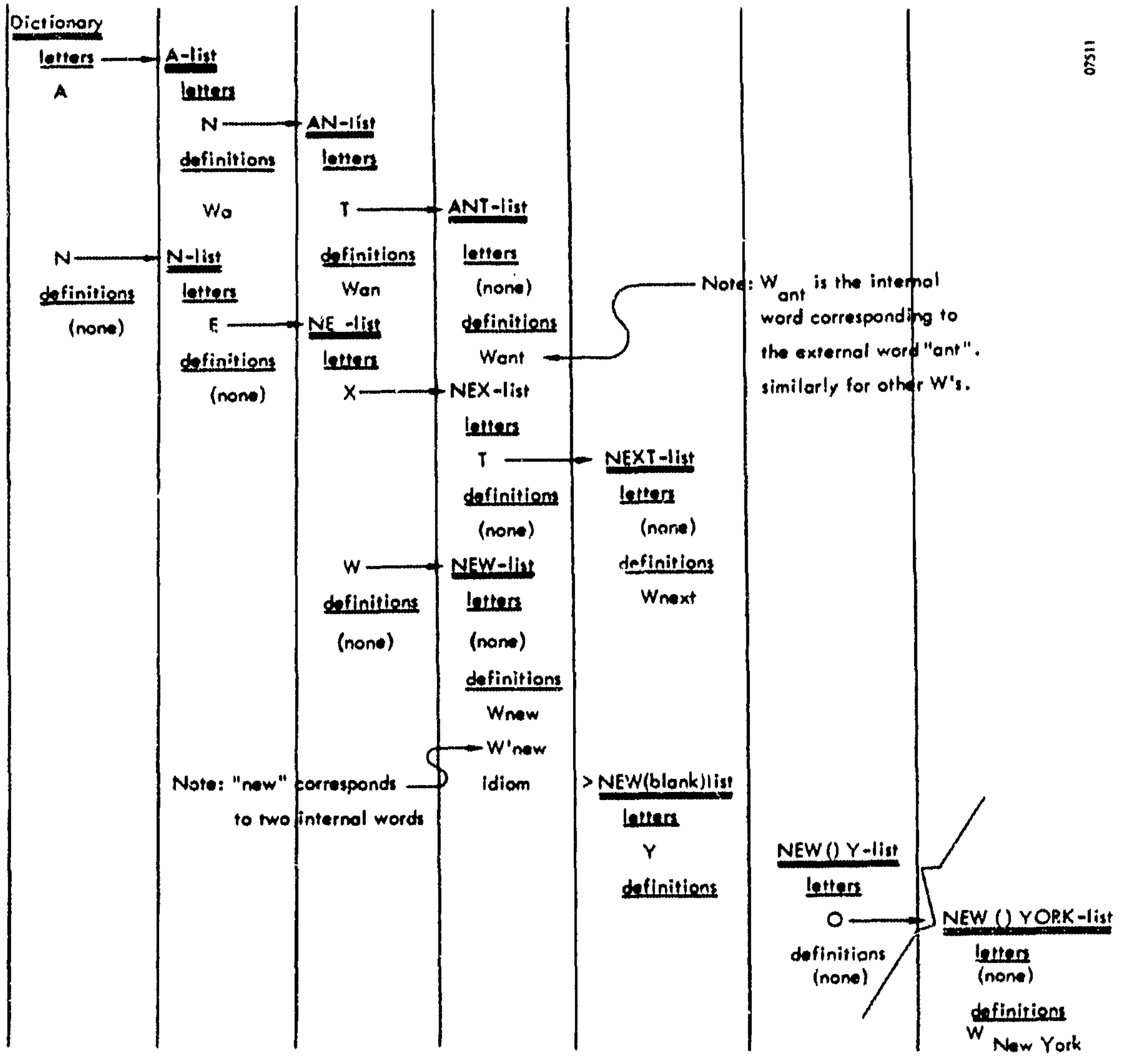


Figure 6. Example of the organization of the DEACON Breadboard Dictionary

Step 2

ANALPR is read in and executed. ANALPR first reads in each dictionary in turn, looking up all words in the sentence and getting all of their corresponding internal words.

Suppose the sentence was: "Where is Black?"

> where: "where" has one internal word, W_{where}
> "is" has two internal words W_{is} and W'_{is}
> (where, say, W_{is} is an F-word, W'_{is} an R-word) .
>
> "Black" has two internal words W_{Black} and W'_{Black} .

ANALPR would then generate four initial subgoals:

> W_{where} W_{is} W_{Black}
>
> W_{where} W_{is} W'_{Black}
>
> W_{where} W'_{is} W_{Black}
>
> W_{where} W'_{is} W'_{Black}

If a word is not in the dictionary, and if it is not found to be part of an idiom, it is printed out exactly as inputted together with the note: "Word(s) not in dictionary." This would be the normal response to a misspelled word.

Example:	Where is Blaak
Output:	Word(s) not in dictionary Blaak

ANALPR then edits in those definitions from Tape 2 that correspond to the internal words used in the initial subgoals.

Step 3

REGIME and Rules are read in and REGIME is executed. REGIME considers each initial subgoal in turn and attempts to apply rules to it. If a rule applies, a new subgoal is formed where the compound phrase created by the rule replaces its constituents. The new subgoal is then worked on. Redundancy checks are made to reduce the proliferation of subgoals. If a subgoal is created that has exactly one phrase, it is put on the Good Analysis List. If a subgoal having more than one phrase is such that no further rule applies, it is abandoned and the last preceding subgoal is considered.

REGIME then edits the Good Analysis List into a more convenient form. It then edits in the semantic transformations which correspond to those rules applied in the subgoals on the Good Analysis List.

An example of the sequence of developing subgoals for a typical sentence may be instructive. The words used are of course their internal forms.

Stage 1:	Smith is an officer.	
Stage 2:	Smith is (an officer) Smith is an officer	[Rule 182 applied to subgoal in stage 1]
Stage 3:	(Smith is) (an officer) Smith is (an officer)	[Rule 99 applied to subgoal in stage 2]
Stage 4:	Smith is (an officer) Smith is an officer	[Subgoal in stage 3 abandoned since no rule applied]
Stage 5:	(Smith is (an officer)) Smith is (an officer) Smith is an officer	[Rule 108 applied to subgoal in stage 4]
Stage 6:	Smith is (an officer) Smith is an officer	[Subgoal in stage 5 put on Good Analysis List]
Stage 7:	Smith is an officer	[subgoal in stage 6 abandoned since no further rule applied]
Stage 8:	(Smith is) an officer Smith is an officer	[Rule 99 applied to subgoal in stage 7]

Stage 9:	Smith is an officer	[Rule 182 was applied to subgoal in Stage 8 but immediately abandoned by redundancy check (see Stage 3). Subgoal of Stage 8 abandoned then when no further rule applied]
Stage 10:		[subgoal of Stage 9 abandoned when no further subgoal applied. Process completed]

Step 4

TRAN and Data are read in and TRAN executed. TRAN works on each element of the Good Analysis List in turn. A given element of the Good Analysis List is a single phrase, usually a compound phrase. Each of its constituents may also be compound. Thus, it has a tree-like structure exhibiting the phrase structure of the underlying sentence. TRAN inverts and linearizes this structure, putting all of the most elemental phrases first, etc. During this process, generators are brought out in front of their corresponding clauses. TRAN then exercises the generators as appropriately scheduled, then carries out the semantic transformations indicated by the rule numbers as they are sequenced in the linear list. The result, of course, is in general the desired answer.

Step 5

ANOUT is read in and executed. For debugging purposes, it is convenient to output the syntactic analysis as well as the answer. Both have been arranged in their respective output lists by TRAN. ANOUT first gathers all the internal words involved in these lists and edits in from Tape 3 the word spellings of the external words to which they correspond. It then develops the appropriate print lines and outputs syntactic analyses and answers.

The following situations may arise, giving the exhibited outputs. (Numbers in the output indicate rules applied.)

1. The input statement is found to be true or false.

```
Example:  The Forrestal is in Boston.

Output:   (  (  The Forrestal )  ( is in Boston ) )
          T  D  F   D             R  R F  V
          99 7                   105

          Answer: —True
```

2. The answer is a single word.

```
Example:  Where is the Forrestal.

Output:   (  Where is  (  the Forrestal ) )
          D  F      F  D  F   D
         156           7

          Answer: —Boston
```

3. The answer is a list.

```
Example:  What ships are in Boston.

Output:   (  (   what  ships ) (  are  in  Boston ) ) )
          T  D   F     L       R  R    F   V
          99 176              105

          Answer: —Shark
                   Forrestal
                   King
```

4. No good syntactic analysis is obtained. This can occur as a result of either of two conditions: either the sentence is not acceptable English or, although acceptable English, is beyond the grammatical capabilities of the system.

```
Example:  Today arrived in the latest list.

Output:   Sentence is beyond the grammatical capability
          of the system.
```

5. No good semantic analysis obtained. Even though the sentence is analyzable syntactically it may not make sense.

```
Example:  The commander of Adm. Black will arrive
          before yesterday.

Output:   Sentence is beyond the grammatical capability
          of the system.
```

6. There may be insufficient data to answer the question. In the particular data base we are using, each ship record contains only one departure and one arrival date. Thus, if a ship arrived in Boston yesterday and was scheduled to depart for Norfolk, with the arrival date in Norfolk already scheduled, there would be insufficient data to establish when it arrived in Boston.

```
Example:  Did the Shark arrive in Boston before yesterday.

Output:   (  (  the  Shark )  (  (   (  did arrive)in Boston)
          T  D  F    D        R  R   R  F   R         F V
          99 7                66 105 60

                                    before (  yesterday)))
                                           N F
          Insufficient Data                78
```

7. Within a sentence there may be a phrase which supposedly describes something but which in matter of fact is a vacuous description. In this case, the sentence ceases to be meaningful, yet the lack of meaning is neither syntactic nor results from general semantic conditions (in contrast to the example in 5). Such a case is: "Adm. Black in New York will depart to Boston." This appears to be a perfectly good sentence. The difficulty is that Adm. Black is not in New York. Thus, the sentence is neither true nor false. This problem has received much attention from logicians. Our approach is straightforward.

```
Example:  Adm. Black in New York will depart to Boston.

Output:   Vacuous description

          (   (  Adm.  Black )  (   in New York ))
          D   D  V     D        V   F  V
          210 2                 215
```

Step 6

The three tapes are rewound, and the routine cycles back to Step 1.

APPENDIX

GRAMMATICAL RULES AND DATA BASE

The DEACON Breadboard rules of grammar are listed in Table 2. The rule numbers are not necessarily consecutive because some rules have been deleted. The Breadboard data base is shown in Figure 7. An expanded version of the data base has also been prepared, which differs from the one shown only in that it includes more task forces, ships, personnel, etc.

Table 2. DEACON Breadboard rules of grammar.

Rule Number	Rule	Rule Number	Rule
1	$V \frown L \Rightarrow L$	17	$A \frown N \Rightarrow V$
2	$V \frown D \Rightarrow D$	18	$V \frown A \Rightarrow X$ $(A_V \neq A)$
3	$V \frown V \Rightarrow L$	20	$\text{At} \frown D \Rightarrow V$
4	$L \frown L \Rightarrow L$	21	$\text{At} \frown V \Rightarrow V$
5	$L \frown V \Rightarrow L$	23	$V \frown \text{of} \frown L \Rightarrow L$
6	$L \frown D \Rightarrow D$	25	$\text{To} \frown V \Rightarrow V$
7	$\text{The} \frown D \Rightarrow D$	29	$\text{On} \frown V \Rightarrow V$
8	$\text{The} \frown L \Rightarrow X$	32	$A \frown \text{of} \frown L \Rightarrow L$
9	$D \ldots X \Rightarrow D \ldots D$	33	$A \frown \text{of} \frown D \Rightarrow V$
10	$\text{The} \frown V \Rightarrow X$ (If V is simple)	34	$D \frown \text{of} \frown L \Rightarrow D$
11	$\text{The} \frown A \Rightarrow X$	35	$L \frown \text{of} \frown L \Rightarrow L$
12	$D \frown A \Rightarrow V$	36	$L \frown \text{Except} \frown V \Rightarrow L$
14	$A \frown V \Rightarrow V$	37	$L \frown \text{Except} \frown L \Rightarrow L$
15	$V \frown A \Rightarrow V$	38	$L \frown \text{Except} \frown D \Rightarrow L$
16	$N \frown A \Rightarrow V$	39	$A \frown \text{Except} \frown V \Rightarrow L$

Table 2. (Cont'd)

Rule Number	Rule	Rule Number	Rule
40	V⁀Except⁀D ⇒ L	77	Tomorrow ⇒ N
41	L⁀Below⁀V = L	78	Yesterday ⇒ N
42	L⁀Above⁀V ⇒ L	79	R . . . Between⁀N⁀and⁀N ⇒ R
43	L From V To V ⇒ L	83	Between⁀N⁀And⁀N . . . R ⇒ R
44	Is . . . To⁀R ⇒ R	87	R . . . From⁀N⁀To⁀N ⇒ R
45	Are . . . To⁀R ⇒ R	91	From⁀N⁀To⁀N . . . R ⇒ R
46	Will . . . R ⇒ R	95	V ⇒ D
47	Is . . . R ⇒ R	96	R . . . From⁀V ⇒ R
48	Are . . . R ⇒ R	97	R . . . To⁀V ⇒ R
49	Will . . . Be⁀R ⇒ R	98	R . . . By⁀N ⇒ R
50	Has . . . R = R	99	D⁀R ⇒ T
51	Was . . . To⁀R ⇒ R	101	N⁀R⁀N ⇒ T
53	Had . . . R ⇒ R	102	V ⇒ N
54	Was . . . To⁀Have⁀R ⇒ R	103	R⁀D ⇒ T
56	Do . . . R ⇒ R	105	R . . . In⁀V ⇒ R
57	Does . . . R ⇒ R	108	D⁀R⁀D ⇒ T
58	Was . . . R ⇒ R	109	D⁀R⁀A ⇒ T
60	Did . . . R ⇒ R	113	R⁀D⁀D ⇒ T
61	Have . . . R ⇒ R	114	R⁀D⁀A ⇒ T
62	R . . . On⁀N ⇒ R	128	L⁀R⁀D ⇒ T
64	On⁀N . . . R ⇒ R	129	L⁀R⁀A ⇒ T
66	R . . . Before⁀N ⇒ R	133	R⁀L⁀D ⇒ T
69	Before⁀N . . . R ⇒ R	134	R⁀L⁀A ⇒ T
71	Today ⇒ N	147	R⁀N⁀N ⇒ T
72	R . . . After⁀N ⇒ R	148	(Number Of)⁀L ⇒ N
75	After⁀N . . . R ⇒ R	149	N⁀L ⇒ D

Table 2. (Cont'd)

Rule Number	Rule	Rule Number	Rule
150	(At Least)⌢N⌢L ⇒ D	176	What⌢L ⇒ D
151	(At Most)⌢N⌢L ⇒ D	177	What⌢A ⇒ D
152	Exactly⌢N⌢L ⇒ D	178	(How Many)⌢L ⇒ D
153	There⌢R ⇒ R	179	(How Many)⌢A ⇒ D
154	R⌢There ⇒ R	180	All⌢L ⇒ D
155	Not⌢R ⇒ R	181	All⌢A ⇒ D
156	What⌢Is⌢D ⇒ D	182	Some⌢L ⇒ D
157	What⌢Is⌢N ⇒ N	183	Some⌢A ⇒ D
158	What⌢Is⌢V ⇒ V	184	N⌢A ⇒ D
159	What⌢Is⌢X ⇒ L	185	(At Least)⌢N⌢A ⇒ D
160	Where⌢Is⌢D ⇒ V	186	(At Most)⌢N⌢A ⇒ D
161	Where⌢Are⌢X ⇒ L	187	L . . . That ⇒ L . . . X
162	Where⌢D⌢Is ⇒ V	188	D⌢T ⇒ D
163	Where⌢X⌢Are ⇒ L	189	X ⇒ D
164	List⌢L ⇒ L	190	L⌢And⌢L ⇒ D
165	List⌢D ⇒ D	191	D⌢And⌢D ⇒ D
166	List⌢N ⇒ N	192	D⌢And⌢L ⇒ D
167	List⌢V ⇒ V	193	L⌢And⌢D ⇒ D
168	List⌢X ⇒ L	194	T⌢And⌢T ⇒ T
169	D⌢Of⌢V ⇒ D	195	L⌢Or⌢L ⇒ D
170	Earliest⌢A ⇒ V	196	D⌢Or⌢D ⇒ D
171	Earliest⌢L ⇒ V	197	D⌢Or⌢L ⇒ D
172	R . . . (By More Than)⌢N ⇒ R	198	L⌢Of⌢D ⇒ D
173	Latest⌢A ⇒ V	199	T⌢Or⌢T ⇒ T
174	Latest⌢L ⇒ V	200	L⌢,⌢L⌢And ⇒ L⌢And
175	L⌢(Ordered By)⌢A ⇒ L	201	D⌢,⌢D⌢And ⇒ L⌢And

Table 2. (Cont'd)

Rule Number	Rule	Rule Number	Rule
202	D⌢,⌢L⌢And ⇒ L⌢And	230	L⌢Period ⇒ L
203	L⌢,⌢D⌢And ⇒ L⌢And	231	D⌢Period ⇒ D
204	T⌢,⌢T⌢And ⇒ L⌢And	232	V⌢Period ⇒ V
205	L⌢,⌢L⌢Or ⇒ L⌢Or	233	N⌢Period ⇒ N
206	D⌢,⌢D⌢Or ⇒ L⌢Or		
207	D⌢,⌢L⌢Or ⇒ L⌢Or		
208	L⌢,⌢D⌢Or ⇒ L⌢Or		
209	T⌢,⌢T⌢Or ⇒ T⌢Or		
210	D⌢V ⇒ D		
211	L⌢A ⇒ L		
213	R⌢(By Less Than) N ⇒ R		
214	What⌢Is⌢L ⇒ L		
215	In⌢V ⇒ V		
217	What⌢Are⌢D ⇒ D		
218	What⌢Are⌢V ⇒ V		
219	What⌢Are⌢X ⇒ L		
220	Where⌢Are⌢D ⇒ V		
221	Where⌢Is⌢X ⇒ L		
222	Where⌢D⌢Are ⇒ V		
223	Where⌢X⌢Is ⇒ L		
224	What⌢Are⌢L ⇒ L		
225	L⌢R ⇒ T		
226	R⌢L ⇒ T		
227	Some⌢V ⇒ Some⌢L		
228	All⌢V ⇒ All⌢L		
229	T⌢Period ⇒ T		

Data Date 8/25/63	Commander	Operational Superior	Admin Superior	Type	Location	Destination	RFS	ETD	ETA	Rank	Date of Rank	Name	Status
6th Fleet	Callan	CNO			Hq.								
TF 60	Callan	6th Flt		TF	Hq.								
TG60.1	McBeth	TF 60		TG	Bos.								
SST 1													
TG 60.2	Gibbons	TF 60		TG	N.Y.								
MMA 15													
DM 24													
TF 61	Black	6th Flt		TF	Bos.								
TG 61.1	Black	6th Flt		TG	CVA 59								
CVA 59													
DD 880													
DD 581													
Personnel													
Headquarters					Norfolk								
Callan		CNO			Hq.					Adm.	1/01/50		Duty
SST 1	McBeth	TG 60.1	SUBDIV	SST	Bos.	Bos.	2	8/27/63	8/30/63			Shark	
McBeth		SST. 1(0)			SST 1					Lt.	8/10/50		Duty
MMA 15	Jones	TG 60.2	SUBDIV	MMA	Sea	N.Y.	0	8/23/63	9/02/63			Fish	
Jones		MMA 15			N.Y.					Lt.	8/12/50		Leave
Smith		MMA 15			MMA 15					Capt.	5/02/52		Duty
DM 24	King	TG 60.2	DESDIV	DM	N.Y.	Bos.	1	8/26/63	8/31/63			Doris Faust	
King		DM 24			DM 24					Cdr.	7/05/52		Duty
Gibbons		DM 24			DM 24					Capt.	2/01/58		Duty
CVA 59	Black	TG 61.1	CARDIV	CVA	Bos.	Norfolk	1	8/27/63	8/31/63			Forrestal	
Black		CVA 59			CVA 59					Adm.	9/05/51		Duty
DD 880	Large	TF 61	DESDIV	DD	Norfolk	N.Y.	3	9/10/63	9/02/63			Doris	
Lange		DD 880			Hq.					Cdr.	10/03/57		Duty
DD 581	Craig	6th Flt	DESDIV	DD	Bos	Bos.	5	7/20/63	8/20/63			King	
Craig		DD 581			Bos.					Cdr.	1/06/50		TDY

07512

Figure 7. DEACON Breadboard data base

REFERENCES

1. Lindsay, R. K. Inferential Memory as the Basis of Machines which Understand Natural Language, Computers and Thought, McGraw-Hill, New York, 1963.

2. Lindsay, R. K. Toward the Development of a Machine which Comprehends, University of Texas, May 1961.

3. Green, B. F., Jr., A. K. Wolf, C. Chomsky, and K. Laughery, Baseball: An Automatic Question Answerer, Computers and Thought, McGraw-Hill, New York, 1963.

4. Wolf, A. K., C. S. Chomsky, and B. F. Green, Jr. The Baseball Program: An Automatic Question-Answerer, Vol. I. Massachusetts Institute of Technology, Lincoln Lab. Tech. Report 306, April 1963.

5. Chomsky, A. N. Syntactic Structures, Mouton & Co., The Hague, The Netherlands, 1957.

6. Chomsky, A. N. Three Models for the Description of Language, IRE Transactions on Information Theory, 1956.

7. Yngve, V. H. Computer Programs for Translation, Sci. American, June, 1962.

8. Tarski, A. The Concept of Truth in Formalized Languages, Logic, Semantics, Metamathematics, Oxford Press, London, 1956.

9. Callan, R. W. and F. B. Thompson, Concept for the Navy Operational Control Complex of the Future, General Electric, TEMPO, RM 62TMP-49, Santa Barbara, California, 1962.

10. Thompson, F. B. Fundamentals Underlying Military Information System Design, General Electric, TEMPO, SP-183, Santa Barbara, California, 1962.

11. Culler, G. H. and B. D. Fried. An Outline Computing Center for Scientific Problems, Proceedings of the Pacific Computer Conference, IEEE, March 1963.

12. Thompson, F. B. The Semantic Interface in Man-Machine Communication, General Electric, TEMPO, RM 63TMP-35, Santa Barbara, California, 1963.

13. Gwynn, J. W. LAP—List Assembly Programming System, General Electric, TEMPO, RM 64TMP-13, Santa Barbara, California, 1964.

14. Craig, J. A., J. Pruett, and F. B. Thompson. DEACON Breadboard Grammar, General Electric, TEMPO, RM 64TMP-14, Santa Barbara, California, 1964.

15. Gibbons, G. D., and F. B. Thompson, DEACON Breadboard Processing, General Electric, TEMPO, RM 64TMP-12, Santa Barbara, California, 1964.

16. Thompson, F. B. The Application and Implementation of DEACON-Type Systems, General Electric, TEMPO, RM 64TMP-11, 1964.